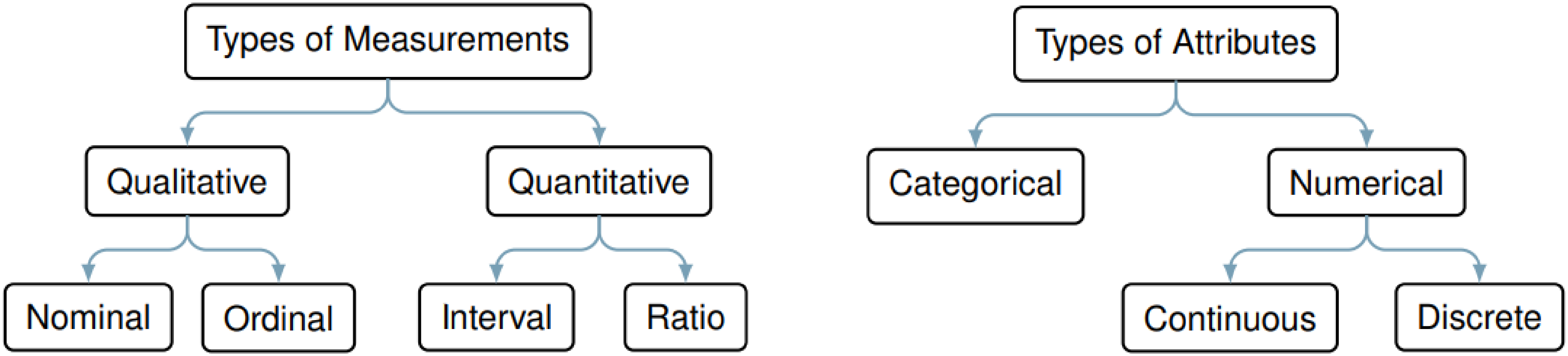
Contents

- Types of Variables
- The measure of Central Tendency
- Encoding Techniques
- Handle NaN Value
- Implementing using Python

Types of Attributes

Two different views:



- *Qualitative* measurements describe an attribute without providing a size or quantity.
- *Quantitative* measurements, often also called *numerical* attributes, are quantitatively measured and often represented in integers or real values.

Nominal:

- Categories, states, or "names of things".
- E.g. $\texttt{hair_color} = \{$auburn, black, blond, brown, grey, red, white$\}$.
- Other examples: `marital_status`, `occupation`, `ID`, `ZIP code`.

Binary:

- Nominal attribute with only two states (0 and 1).
- **Symmetric binaries**: both outcomes equally important, such as `sex`.
- **Asymmetric binary**: outcomes not equally important.
  E.g. medical test (positive vs. negative).
  Convention: assign 1 to most important outcome (e.g. diabetes, HIV positive).

Ordinal:

- Values have a meaningful order (ranking), but magnitude between successive values is not known.
- E.g. $\texttt{size} = \{$small, medium, large$\}$, grades, army rankings.

Continuous Attributes

- Has real numbers as attribute values.
  E.g. temperature, height, or weight.
- Practically, real values can only be measured and represented using a finite number of digits.
- Continuous attributes are typically represented as floating-point variables.

Discrete Attributes

- Has finite or countably infinite elements.
  E.g. ZIP code, profession, or the set of words in a collection of documents.
- Sometimes represented as integer variables.

Note

Binary attributes are a special case of discrete attributes.

Measures of Central Tendency

Mean, Median, Mode

- **Mean:** The mean is the most popular and well known measure of central tendency.

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

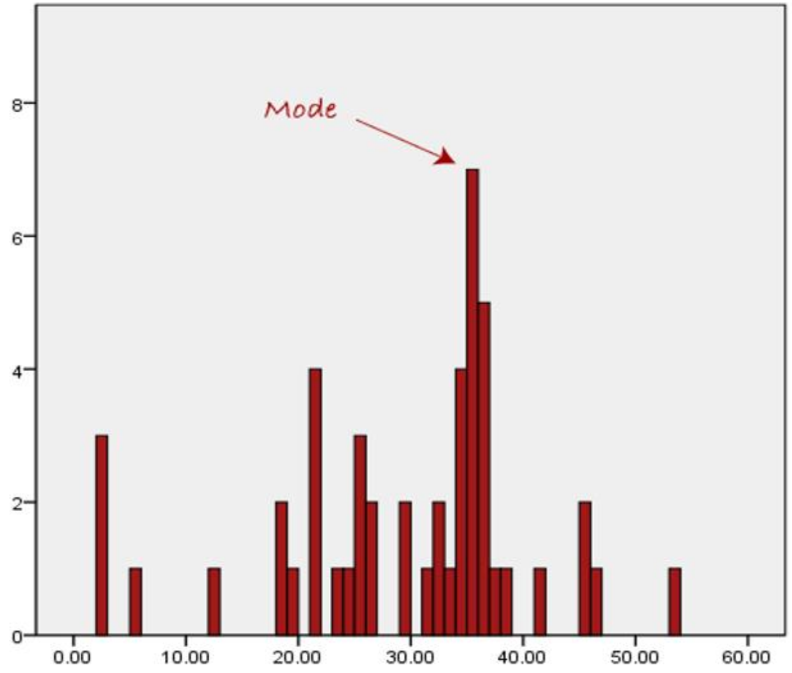
- **Median:** The median is the middle score for a set of data that has been arranged in order of magnitude.

65	55	89	56	35	14	56	55	87	45	92

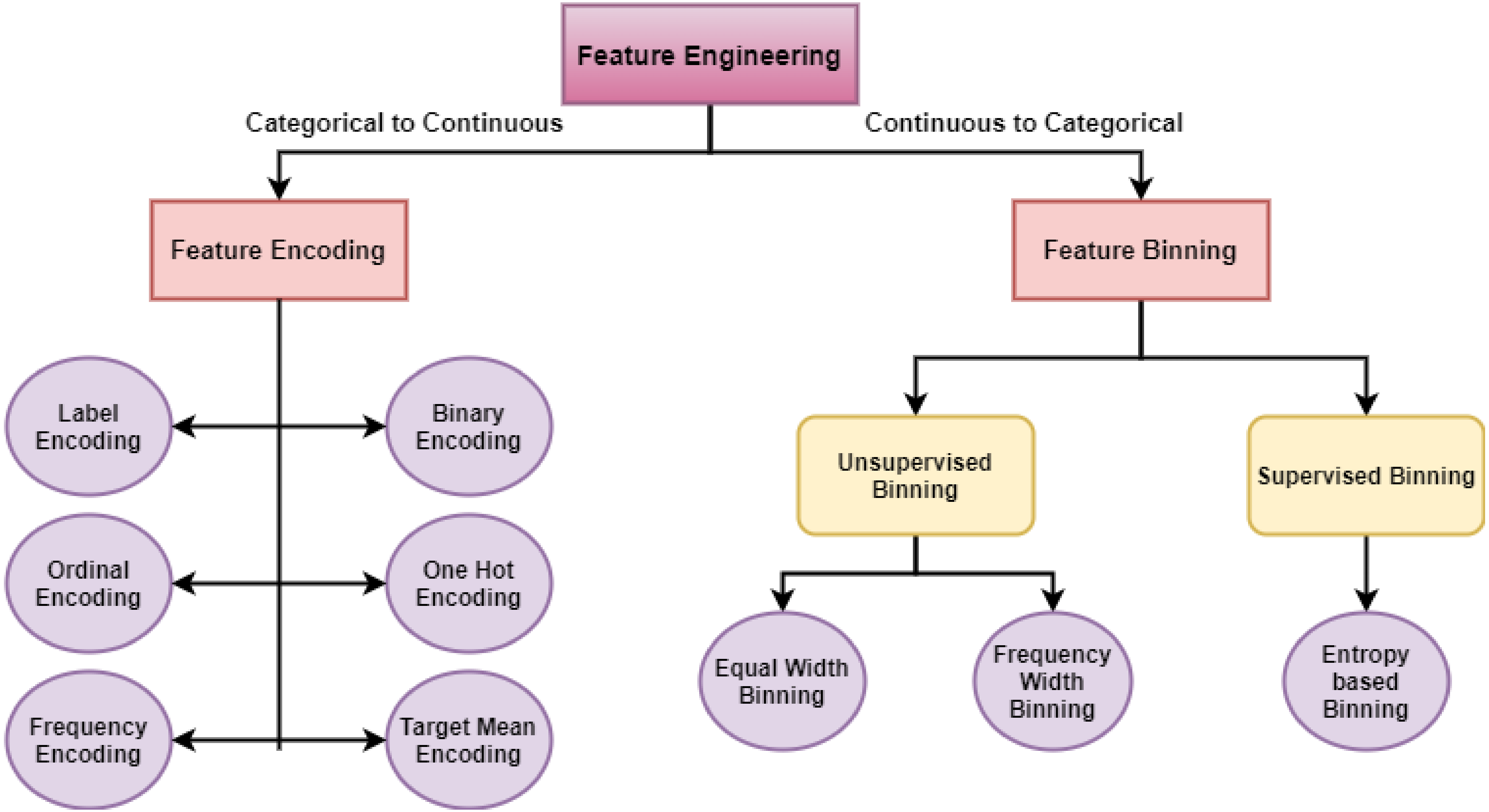
We first need to rearrange that data into order of magnitude (smallest first):

14	35	45	55	55	**56**	56	65	87	89	92

- **Mode:** The mode is the most frequent score in our data set.



Encoding in ML



Encoding in ML

Encoding is a technique of ***converting categorical variables into numerical values*** so that they can be easily fitted to a machine learning model. Since ***most algorithms work better with numerical inputs***, encoding is a crucial step for preparing and representing data, especially when dealing with non-numerical data types like ***categorical or textual data.***

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

Encoding in ML

Encoding is a technique of ***converting categorical variables into numerical values*** so that they can be easily fitted to a machine learning model. Since ***most algorithms work better with numerical inputs***, encoding is a crucial step for preparing and representing data, especially when dealing with non-numerical data types like ***categorical or textual data.***

Colour
Green
Red
Blue

Green	Red	Blue
0	1	1
1	1	1
1	0	1
0	0	0
0	1	0

Types of Encoding

Some common encoding techniques in machine learning:

1. **Label Encoding**
2. One-Hot Encoding
3. Binary Encoding
4. Ordinal Encoding
5. Frequency Encoding
6. Mean Encoding
7. Embedding

❖ **Label Encoding:**

Label encoding assigns each **unique category value a numerical code**. It is straightforward but introduces a new problem: the model might infer a natural ordering in categories, which might not be intended. For example: ["red" < "blue" < "green"] to [0, 1, 2]

- **Advantages**:
  - Simple to implement and keeps the dataset's dimensionality unchanged.
  - Useful for ordinal data or tree-based models that can handle ordinality.
- **Disadvantages**:
  - Imposes an ordinal relationship where it might not exist, potentially leading to poor model performance for non-ordinal data.
  - Not suitable for linear models unless the data is ordinal.

❖ **Label Encoding:**

Label encoding assigns each **unique category value a numerical code**. It is straightforward but introduces a new problem: the model might infer a natural ordering in categories, which might not be intended. For example: ["red" < "blue" < "green“] to [0, 1, 2]

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

→

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

Types of Encoding

Some common encoding techniques in machine learning:

1. Label Encoding
2. **One-Hot Encoding**
3. Binary Encoding
4. Ordinal Encoding
5. Frequency Encoding
6. Mean Encoding
7. Embedding

❖ One-Hot Encoding:

One-hot encoding transforms **each category value into a new binary column and assigns a 1 or 0** (presence or absence) value to the column. This method is widely used for nominal categories without intrinsic ordering.

Feature (Color)
Red
Green
Yellow
Green
Red

One Hot Encoding →

One Hot Encoded Vector	Red	Green	Yellow
[1,00]	1	0	0
[0,1,0]	0	1	0
[0,0,1]	0	0	1
[0,1,0]	0	1	0
[1,00]	1	0	0

❖ **One-Hot Encoding:**

One-hot encoding transforms **each category value into a new binary column and assigns a 1 or 0** (presence or absence) value to the column. This method is widely used for nominal categories without intrinsic ordering.

Feature (Color)
Red
Green
Yellow
Green
Red

→ One Hot Encoding

Red	Green
1	0
0	1
0	0
0	1
1	0

Yellow Column dropped to avoid the Dummy Variable Trap

❖ **One-Hot Encoding:**

One-hot encoding transforms **each category value into a new binary column and assigns a 1 or 0** (presence or absence) value to the column. This method is widely used for nominal categories without intrinsic ordering.

- **Advantages:**
  - Eliminates any ordinal relationship, making it suitable for nominal data.
  - Easy to understand and implement.
- **Disadvantages:**
  - Can lead to a high-dimensional feature space, increasing memory and computational costs.
  - Not suitable for high cardinality features (many unique values).

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Types of Encoding

Some common encoding techniques in machine learning:

1. Label Encoding
2. One-Hot Encoding
3. **Binary Encoding**
4. Ordinal Encoding
5. Frequency Encoding
6. Mean Encoding
7. Embedding

❖ **Binary Encoding:**

Binary encoding **first converts categories into numerical labels** and **then transforms those labels into binary codes**. Each binary digit gets its column. This method can be **more efficient than one-hot encoding when dealing with many categories.**

Temperature	Order	Binary	Temperature_0	Temperature_1	Temperature_2
Hot	1	001	0	0	1
Cold	2	010	0	1	0
Very Hot	3	011	0	1	1
Warm	4	100	1	0	0
Hot	1	001	0	0	1
Warm	4	100	1	0	0
Warm	4	100	1	0	0
Hot	1	001	0	0	1
Hot	1	001	0	0	1
Cold	2	010	0	1	0

❖ **Binary Encoding:**

Binary encoding **first converts categories into numerical labels** and **then transforms those labels into binary codes**. Each binary digit gets its column. This method can be **more efficient than one-hot encoding when dealing with many categories.**

- **Advantages**:
  - Reduces the dimensionality compared to one-hot encoding for high cardinality features.
  - Retains more information in fewer dimensions than label encoding.
- **Disadvantages**:
  - More complex and harder to interpret than one-hot or label encoding.
  - Binary representation can introduce relationships that do not exist in the original categorical data.

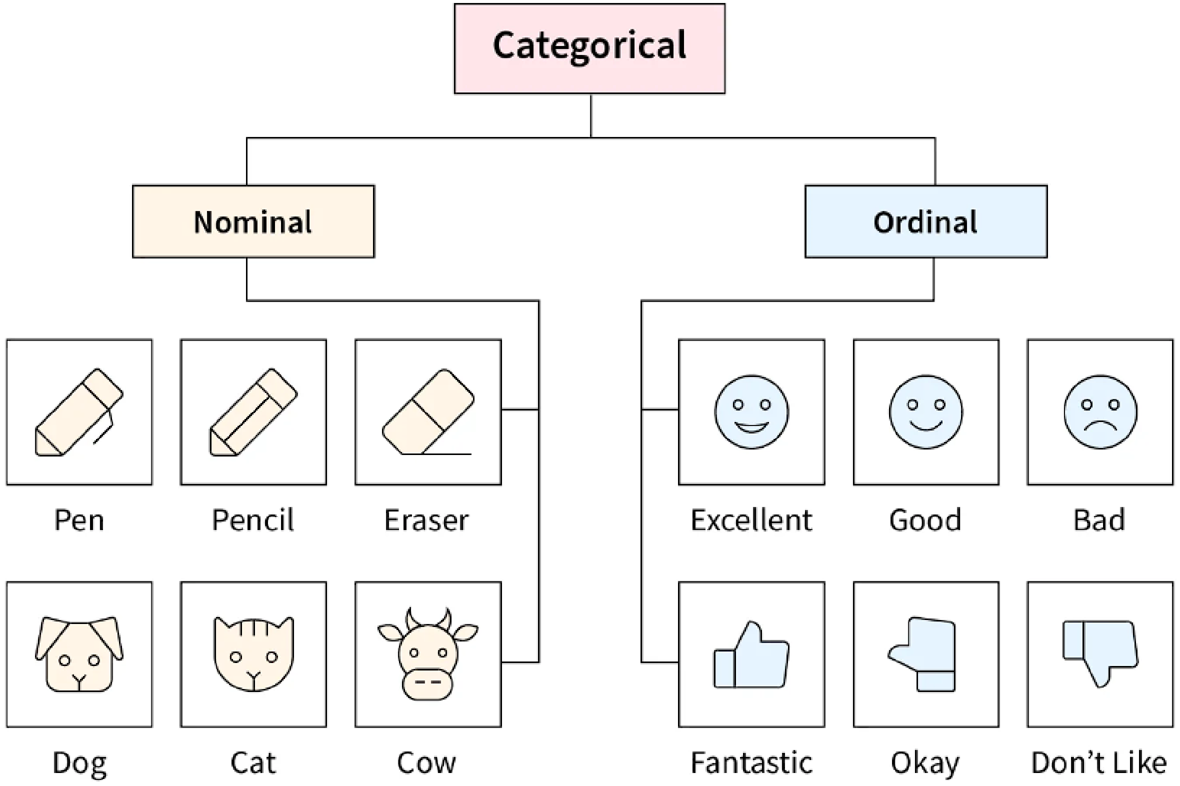
Types of Encoding

Some common encoding techniques in machine learning:

1. Label Encoding
2. One-Hot Encoding
3. Binary Encoding
4. **Ordinal Encoding**
5. Frequency Encoding
6. Mean Encoding
7. Embedding

❖ **Ordinal Encoding:**

Ordinal encoding is like label encoding but specifically applies to ordinal data where the order of categories is important (Example: "low" < "medium" < "high"). The categories are converted into an ordered numerical scale.



❖ **Ordinal Encoding:**

Ordinal encoding is like label encoding but specifically applies to ordinal data where the order of categories is important (Example: "low" < "medium" < "high"). The categories are converted into an ordered numerical scale.

Instances: [extra large -> large -> medium -> small]
Encoded data: [0, 1, 2, 3]

cost	size	size_endoced
50	large	1.0
35	small	3.0
75	extra large	0.0
42	medium	2.0
54	large	1.0
71	extra large	0.0

❖ **Ordinal Encoding:**

Ordinal encoding is like label encoding but specifically applies to ordinal data where the order of categories is important (Example: "low" < "medium" < "high"). The categories are converted into an ordered numerical scale.

- **Advantages:**
  - Directly encodes the order of the categories, making it suitable for ordinal data.
  - Keeps the dataset's dimensionality unchanged.
- **Disadvantages:**
  - Incorrect use on nominal data can introduce artificial ordinality, leading to misleading results.
  - The choice of encoding values can impact model performance.

Types of Encoding

Some common encoding techniques in machine learning:

1. Label Encoding
2. One-Hot Encoding
3. Binary Encoding
4. Ordinal Encoding
5. **Frequency Encoding**
6. Mean Encoding
7. Embedding

Encoding in ML

Types of Encoder

❖ **Frequency Encoding:**

Frequency encoding replaces categories with their frequencies or counts. This approach can help algorithms to understand the prominence of certain categories over others.

$$FrequencyEncoding = \frac{frequency(category)}{size(data)}$$

Numerical value	Animal
1.5	cat
3.6	cat
42	dog
7.1	crocodile

Frequency encoding →

Numerical value	Animal_freq
1.5	0.5
3.6	0.5
42	0.25
7.1	0.25

❖ **Frequency Encoding:**

Frequency encoding replaces categories with their frequencies or counts. This approach can help algorithms to understand the prominence of certain categories over others.

- **Advantages**:
  - Handles high cardinality data well by using frequencies, which can be more informative than labels.
  - Can capture the importance of category frequency for the prediction task.
- **Disadvantages**:
  - Loses information about the category itself, only retaining frequency.
  - Similar frequencies can lead to collisions where different categories are represented by the same value.

❖ **Mean Encoding:**

Mean encoding replaces categorical values with the **mean target value** for that category. Mean encoding is particularly useful for categorical features with a high number of unique categories. It's particularly useful for high cardinality data and can help in tree-based algorithms.

❖ Mean Encoding:

Mean encoding replaces categorical values with the **mean target value** for that category. Mean encoding is particularly useful for categorical features with a **high number of unique categories**. It's particularly useful for **high cardinality data** and can help in **tree-based algorithms**.

- **Advantages**:
  - Incorporates target information, which can improve model performance.
  - Efficient representation for high cardinality features.
- **Disadvantages**:
  - Prone to overfitting, especially with small dataset sizes or categories with few instances.
  - Requires careful regularisation or validation techniques to prevent data leakage.

❖ **Embedding:**

Embeddings are a sophisticated way to represent categories in high-dimensional spaces. This technique is often used in deep learning for handling textual data, where each word or category is represented by a dense vector of floating-point numbers.

❖ **Embedding:**
Embeddings are a sophisticated way to represent categories in high-dimensional spaces. This technique is often used in deep learning for handling textual data, where each word or category is represented by a dense vector of floating-point numbers.

- **Advantages**:
  - Captures complex relationships and patterns in high-dimensional space, especially useful for deep learning models and NLP.
  - Reduces dimensionality while retaining the richness of data, suitable for high cardinality and textual data.
- **Disadvantages**:
  - Requires significant computational resources and data to train effectively.
  - More complex and harder to interpret than other encoding techniques.

Thank you!